Convex Graph Invariants

Venkat Chandrasekaran, Pablo A. Parrilo, and Alan S. Willsky *

Laboratory for Information and Decision Systems
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

December 2, 2010

Abstract

The structural properties of graphs are usually characterized in terms of invariants, which are functions of graphs that do not depend on the labeling of the nodes. In this paper we study convex graph invariants, which are graph invariants that are convex functions of the adjacency matrix of a graph. Some examples include functions of a graph such as the maximum degree, the MAXCUT value (and its semidefinite relaxation), and spectral invariants such as the sum of the k largest eigenvalues. Such functions can be used to construct convex sets that impose various structural constraints on graphs, and thus provide a unified framework for solving a number of interesting graph problems via convex optimization. We give a representation of all convex graph invariants in terms of certain elementary invariants, and describe methods to compute or approximate convex graph invariants tractably. We also compare convex and non-convex invariants, and discuss connections to robust optimization. Finally we use convex graph invariants to provide efficient convex programming solutions to graph problems such as the deconvolution of the composition of two graphs into the individual components, hypothesis testing between graph families, and the generation of graphs with certain desired structural properties.

Keywords: Graphs; graph invariants; convex optimization; spectral invariants; majorization; robust optimization; graph deconvolution; graph sampling; graph hypothesis testing

1 Introduction

Graphs are useful in many applications throughout science and engineering as they offer a concise model for relationships among a large number of interacting entities. These relationships are often best understood using structural properties of graphs. *Graph invariants* play an important role in characterizing abstract structural features of a graph, as they do not depend on the labeling of the nodes of the graph. Indeed families of graphs that share common structural attributes are often specified via graph invariants. For example bipartite graphs can be defined by the property that they contain no cycles of odd length, while the family of regular graphs consists of graphs in which all nodes have the same degree. Such descriptions of classes of graphs in terms of invariants have

---

1. REPORT DATE **02 DEC 2010**	2. REPORT TYPE	3. DATES COVERED **00-00-2010 to 00-00-2010**
4. TITLE AND SUBTITLE **Convex Graph Invariants**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,Laboratory for Information and Decision Systems,Department of Electrical Engineering and Computer Science,Cambridge,MA,02139**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
To be presented at the 2010 CMS (Canadian Mathematical Society) Winter Meeting, Dec 4-6, 2010Vancouver, British Columbia

14. ABSTRACT
The structural properties of graphs are usually characterized in terms of invariants, which are functions of graphs that do not depend on the labeling of the nodes. In this paper we study convex graph invariants, which are graph invariants that are convex functions of the adjacency matrix of a graph. Some examples include functions of a graph such as the maximum degree the MAXCUT value (and its semide nite relaxation), and spectral invariants such as the sum of the k largest eigenvalues. Such functions can be used to construct convex sets that impose various structural constraints on graphs, and thus provide a uni ed framework for solving a number of interesting graph problems via convex optimization. We give a representation of all convex graph invariants in terms of certain elementary invariants, and describe methods to compute or approximate convex graph invariants tractably. We also compare convex and non-convex invariants, and discuss connections to robust optimization. Finally we use convex graph invariants to provide e cient convex programming solutions to graph problems such as the deconvolution of the composition of two graphs into the individual components, hypothesis testing between graph families, and the generation of graphs with certain desired structural properties.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**30**	

found applications in areas as varied as combinatorics [15], network analysis in chemistry [8] and in biology [31], and in machine learning [27]. For instance the treewidth [34] of a graph is a basic invariant that governs the complexity of various algorithms for graph problems.

We begin by introducing three canonical problems involving structural properties of graphs, and the development of a unified solution framework to address these questions serves as motivation for our discussion throughout this paper.

- **Graph deconvolution.** Suppose we are given a graph that is the combination of two known graphs overlaid on the same set of nodes. How do we recover the individual components from the composite graph? For example in Figure 1 we are given a composite graph that is formed by adding a cycle and the Clebsch graph. Given no extra knowledge of any labeling of the nodes, can we "deconvolve" the composite graph into the individual cycle/Clebsch graph components?

- **Graph generation.** Given certain structural constraints specified by invariants how do we produce a graph that satisfies these constraints? A well-studied example is the question of constructing expander graphs. Another example may be that we wish to recover a graph given constraints, for instance, on certain subgraphs being forbidden, on the degree distribution, and on the spectral distribution.

- **Graph hypothesis testing.** Suppose we have two families of graphs, each characterized by some common structural properties specified by a set of invariants; given a new sample graph which of the two families offers a "better explanation" of the sample graph (see Figure 2)?

In Section 2 we describe these problems in more detail, and also give some concrete applications in network analysis and modeling in which such questions are of interest.

To efficiently solve problems such as these we wish to develop a collection of tractable computational tools. Convex relaxation techniques offer a candidate framework as they possess numerous favorable properties. Due to their powerful modeling capabilities, convex optimization methods can provide tractable formulations for solving difficult combinatorial problems exactly or approximately. Further convex programs may often be solved effectively using general-purpose off-the-shelf software. Finally one can also give conditions for the success of these convex relaxations based on standard optimality results from convex analysis.

Motivated by these considerations we introduce and study *convex graph invariants* in Section 3. These invariants are convex functions of the adjacency matrix of a graph. More formally letting A denote the adjacency matrix of a (weighted) graph, a convex graph invariant is a convex function f such that $f(A) = f(\Pi A \Pi^T)$ for all permutation matrices Π. Examples include functions of a graph such as the maximum degree, the MAXCUT value (and its semidefinite relaxation), the second smallest eigenvalue of the Laplacian (a concave invariant), and spectral invariants such as the sum of the k largest eigenvalues; see Section 3.3 for a more comprehensive list. As some of these invariants may possibly be hard to compute, we discuss in the sequel the question of approximating intractable convex invariants. We also study *invariant convex sets*, which are convex sets with the property that a symmetric matrix A is a member of such a set if and only if $\Pi A \Pi^T$ is also a member of the set for all permutations Π. Such convex sets are useful in order to impose various structural constraints on graphs. For example invariant convex sets can be used to express forbidden subgraph constraints (i.e., that a graph does not contain a particular subgraph such as a triangle), or require that a graph be connected; see Section 3.4 for more examples. We compare the strengths and weaknesses of convex graph invariants versus more general non-convex graph invariants. Finally we also provide a robust optimization perspective of invariant convex sets. In particular we make

connections between our work and the data-driven perspective on robust optimization studied in [6].

In order to systematically evaluate the expressive power of convex graph invariants we analyze *elementary* convex graph invariants, which serve as a basis for constructing arbitrary convex invariants. Given a symmetric matrix P, these elementary invariants (again, possibly hard to compute depending on the choice of P) are defined as follows:

$$\Theta_P(A) = \max_{\Pi} \text{Tr}(P\Pi A\Pi^T), \tag{1}$$

where A represents the adjacency matrix of a graph, and the maximum is taken over all permutation matrices Π. It is clear that Θ_P is a convex graph invariant, because it is expressed as the maximum over a set of linear functions. Indeed several simple convex graph invariants can be expressed using functions of the form (1). For example $P = I$ gives us the total sum of the node weights, while $P = \mathbf{1}\mathbf{1}^T - I$ gives us twice the total (weighted) degree. Our main theoretical results in Section 3 can be summarized as follows: First we give a representation theorem stating that any convex graph invariant can be expressed as the supremum over elementary convex graph invariants (1) (see Theorem 3.4). Second we have a similar result stating that any invariant convex set can be expressed as the intersection of convex sets given by level sets of the elementary invariants (1) (see Proposition 3.6). These results follow as a consequence of the separation theorem from convex analysis. Finally we also show that for any two non-isomorphic graphs given by adjacency matrices A_1 and A_2, there exists a P such that $\Theta_P(A_1) \neq \Theta_P(A_2)$ (see Lemma 3.10). Hence convex graph invariants offer a *complete* set of invariants as they can distinguish between non-isomorphic graphs.

In Section 3.7 we discuss an important subclass of convex graph invariants, namely the set of convex *spectral invariants*. These are convex functions of symmetric matrices that depend only on the eigenvalues, and can equivalently be expressed as the set of convex functions of symmetric matrices that are invariant under conjugation by orthogonal matrices (note that convex graph invariants are only required to be invariant with respect to conjugation by permutation matrices) [13]. The properties of convex spectral invariants are well-understood, and they are useful in a number of practically relevant problems (e.g., characterizing the subdifferential of a unitarily invariant matrix norm [39]). These invariants play a prominent role in our experimental demonstrations in Section 5.

As noted above convex graph invariants, and even elementary invariants, may in general be hard to compute. In Section 4 we investigate the question of approximately computing these invariants in a tractable manner. For many interesting special cases such as the MAXCUT value of a graph, or (the inverse of) the stability number, there exist well-known tractable semidefinite programming (SDP) relaxations that can be used as surrogates instead [22, 32]. More generally functions of the form of our elementary convex invariants (1) have appeared previously in the literature; see [11] for a survey. Specifically we note that evaluating the function $\Theta_P(A)$ for any fixed A, P is equivalent to solving the so-called Quadratic Assignment Problem (QAP), and thus we can employ various tractable linear programming, spectral, and SDP relaxations of QAP [40, 11, 33]. In particular we discuss recent work [14] on exploiting group symmetry in SDP relaxations of QAP, which is useful for approximately computing elementary convex graph invariants in many interesting cases.

Finally in Section 5 we return to the motivating problems described previously, and give solutions to these questions. These solutions are based on convex programming formulations, with convex graph invariants playing a fundamental role. We give theoretical conditions for the success of these convex formulations in solving the problems discussed above, and experimental demonstration for their effectiveness in practice. Indeed the framework provided by convex graph invariants allows for a *unified* investigation of our proposed solutions. As an example result we give a tractable

convex program (in fact an SDP) in Section 5.2 to "deconvolve" the cycle and the Clebsch graph from a composite graph consisting of these components (see Figure 1); a salient feature of this convex program is that it only uses *spectral invariants* to perform the decomposition.

Summary of contributions We emphasize again the main contributions of this paper. We begin by introducing three canonical problems involving structural properties of graphs. These problems arise in various applications (see Section 2), and serve as a motivation for our discussion in this paper. In order to solve these problems we introduce convex graph invariants, and investigate their properties (see Section 3). Specifically we provide a representation theorem of convex graph invariants in terms of elementary invariants, and we make connections between these ideas and concepts from other areas such as robust optimization. Finally we describe tractable convex programming solutions to the motivating problems based on convex graph invariants (see Section 5). Therefore, convex graph invariants provide a useful computational framework based on convex optimization for graph problems.

Related previous work We note that convex optimization methods have been used previously to solve various graph-related problems. We would particularly like to emphasize a body of work on convex programming formulations to optimize convex functions of the Laplacian eigenvalues of graphs [10, 9] subject to various constraints. Although our objective is similar in that we seek solutions based on convex optimization to graph problems, our work is different in several respects from these previous approaches. While the problems discussed in [9] explicitly involved the optimization of spectral functions, other graph problems such as those described in Section 2 may require non-spectral approaches (for example, hypothesis testing between two families of graphs that are isospectral, i.e., have the same spectrum, but are distinguished by other structural properties). As convex spectral invariants form a subset of convex graph invariants, the framework proposed in this paper offers a larger suite of convex programming methods for graph problems. More broadly our work is the first to formally introduce and characterize convex graph invariants, and to investigate their properties as natural mathematical objects of independent interest.

Outline In Section 2 we give more details of the questions that motivate our study of convex graph invariants. Section 3 gives the definition of convex graph invariants and invariant convex sets, as well as several examples of these such functions and sets. We also discuss various properties of convex graph invariants in this section. In Section 4 we investigate the question of efficiently computing approximations to intractable convex graph invariants. We give detailed solutions using convex graph invariants to each of our motivating problems in Section 5, and we conclude with a brief discussion in Section 6.

2 Applications

In this section we describe three problems involving structural properties of graphs, which serve as a motivation for our investigation of convex graph invariants. In Section 5 we give solutions to these problems using convex graph invariants.

2.1 Graph Deconvolution

Suppose we are given a graph that is formed by overlaying two graphs on the same set of nodes. More formally we have a graph whose adjacency matrix is formed by adding the adjacency matrices
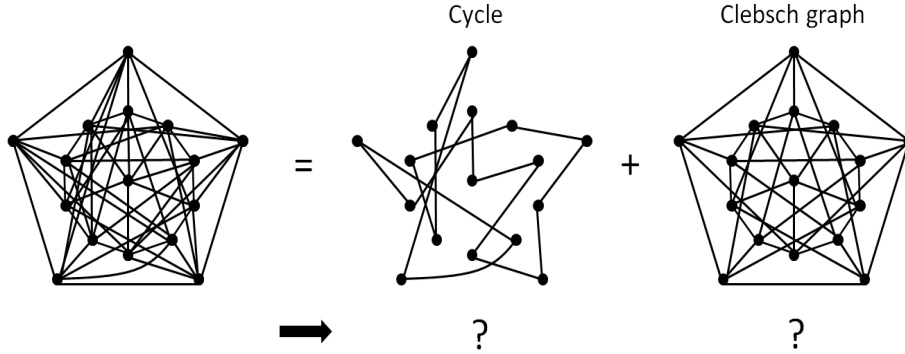
Figure 1: An instance of a deconvolution problem: Given a composite graph formed by adding the 16-cycle and the Clebsch graph, we wish to recover the individual components. The Clebsch graph is an example of a strongly regular graph on 16 nodes [21]; see Section 5.2 for more details about the properties of such graphs.

of two known graphs. However, we do not have any information about the relative labeling of the nodes in the two component graphs. Can we recover the individual components from the composite graph? As an example suppose we are given the combination of a cycle and a grid, or a cycle and the Clebsch graph, on the same set of nodes. Without any additional information about the labeling of the nodes, which may reveal the cycle/grid or cycle/Clebsch graph structure, the goal is to recover the individual components. Figure 1 gives a graphical illustration of this question. In general such decomposition problems may be ill-posed, and it is of interest to give conditions under which unique deconvolution is possible as well as to provide tractable computational methods to recover the individual components. In Section 5.2 we describe an approach based on convex optimization for graph deconvolution; for example this method decomposes the cycle and the Clebsch graph from a composite graph consisting of these components (see Figure 1) using only the spectral properties of the two graphs.

Well-known problems that have the flavor of graph deconvolution include the *planted clique* problem, which involves identifying hidden cliques embedded inside a larger graph, and the *clustering* problem in which the goal is to decompose a large graph into smaller densely connected clusters by removing just a few edges. Convex optimization approaches for solving such problems have been proposed recently [1, 2]. Graph deconvolution more generally may include other kinds of embedded structures beyond cliques.

Applications of graph deconvolution arise in network analysis in which one seeks to better understand a complex network by decomposing it into simpler components. Graphs play an important role in modeling, for example, biological networks [31] and social networks [25, 18], and lead to natural graph deconvolution problems in these areas. For instance graphs are useful for describing social exchange networks of interactions of multiple agents, and graph decompositions are useful for describing the structure of optimal bargaining solutions in such networks [26]. In a biological network setting, transcriptional regulatory networks of bacteria have been observed to consist of small subgraphs with specific structure (called motifs) that are connected together using a "backbone" [16]. Decomposing such regulatory networks into the component structures is useful for obtaining a better understanding of the high-level properties of the composite network.
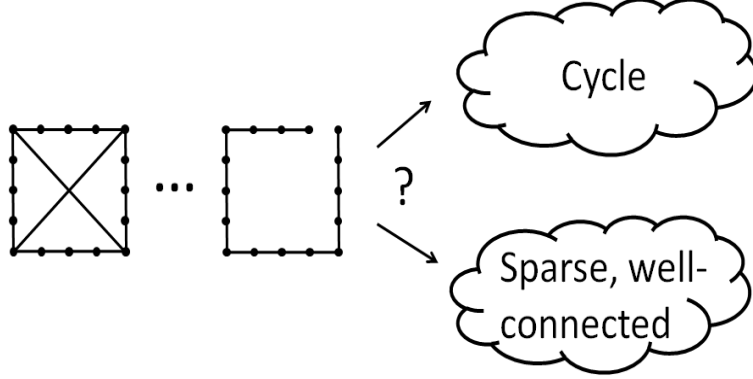
Figure 2: An instance of a hypothesis testing problem: We wish to decide which family of graphs offers a "better explanation" for a given candidate sample graph.

2.2 Generating Graphs with Desired Structural Properties

Suppose we wish to construct a graph with certain prescribed structural constraints. A very simple example may be the problem of constructing a graph in which each node has degree equal to two. A graph given by a single cycle satisfies this constraint. A less trivial problem is one in which the objective may be to build a connected graph with constraints on the spectrum of the adjacency matrix, the degree distribution, and the additional requirements that the graph be triangle-free and square-free. Of course such graph reconstruction problems may be infeasible in general, as there may be no graph consistent with the given constraints. Therefore it is of interest to derive suitable conditions under which this problem may be well-posed, and to develop a suitably flexible yet tractable computational framework to incorporate any structural information available about a graph.

A prominent instance of a graph construction problem that has received much attention is the question of generating expander graphs [24]. Expanders are, roughly speaking, sparse graphs that are well-connected, and they have found applications in numerous areas of computer science. Methods used to construct expanders range from random sampling approaches to deterministic constructions based on Ramanujan graphs [24]. In Section 5.3 we describe an approach based on convex optimization to generate sparse, weighted graphs with small degree and large spectral gap.

2.3 Graph Hypothesis Testing

As our third problem we consider a more statistically motivated question. Suppose we have two families of graphs each characterized by some common structural properties specified by certain invariants. Given a new sample graph which of these two families offers a "better explanation" for the sample graph? For example as illustrated in Figure 2 we may have two families of graphs – one being the collection of cycles, and the other being the set of sparse, well-connected graphs. If a new sample graph is a path (i.e., a cycle with an edge removed), we would expect that the family of cycles should be a better explanation. On the other hand if the sample is a cycle plus some edges connecting diametrically opposite nodes, then the second family of sparse, well-connected graphs offers a more plausible fit. Notice that these classes of graphs may often be specified in terms of *different* sets of invariants, and it is of interest to develop a suitable framework in which we can incorporate diverse structural information provided about graph families.

We differentiate this problem from the well-studied question of *testing properties* of graphs [23].

6

Examples of property testing include testing whether a graph is 3-colorable, or whether it is close to being bipartite. An important goal in property testing is that one wishes to test for graph properties by only making a small number of "queries" of a graph. We do not explicitly seek such an objective in our algorithms for hypothesis testing. We also note that hypothesis testing can be posed more generally than a yes/no question as in property testing, and as mentioned above the two families in hypothesis testing may be specified in terms of very different sets of invariants.

In order to address the hypothesis testing question in a statistical framework, we would need a statistical theory for graphs and appropriate error metrics with respect to which one could devise optimal decision rules. In Section 5.4 we discuss a computational approach to the hypothesis testing problem using convex graph invariants that gives good empirical performance, and we defer the issue of developing a formal statistical framework to future work.

3   Convex Graph Invariants

In this section we define convex graph invariants, and discuss their properties. Throughout this paper we denote the space of $n \times n$ symmetric matrices by $\mathbf{S}^n \simeq \mathbb{R}^{\binom{n+1}{2}}$. All our definitions of convexity are with respect to the space $\mathbf{S}^n$. We consider undirected graphs that do not have multiple edges and no self-loops; these are represented by adjacency matrices that lie in $\mathbf{S}^n$. Therefore a graph may possibly have node weights and edge weights. A graph is said to be *unweighted* if its node weights are zero, and if each edge has a weight of one (non-edges have a weight of zero); otherwise a graph is said to be *weighted*. Let $\mathbf{e}_i \in \mathbb{R}^n$ denote the vector with a one in the i'th entry and zero elsewhere, let I denote the $n \times n$ identity matrix, let $\mathbf{1} \in \mathbb{R}^n$ denote the all-ones vector, and let $J = \mathbf{1}\mathbf{1}^T \in \mathbf{S}^n$ denote the all-ones matrix. Further we let $\mathcal{A} = \{A \ : \ A \in \mathbf{S}^n, \ 0 \le A_{i,j} \le 1 \ \forall i, j\}$; we will sometimes find it useful in our examples in Section 3.4 to restrict our attention to graphs with adjacency matrices in $\mathcal{A}$. Next let $\mathrm{Sym}(n)$ denote the symmetric group over n elements, i.e., the group of permutations of n elements. Elements of this group are represented by $n \times n$ permutation matrices. Let $\mathrm{O}(n)$ represent the orthogonal group of $n \times n$ orthogonal matrices. Finally given a vector $\mathbf{x} \in \mathbb{R}^n$ we let $\overline{\mathbf{x}}$ denote the vector obtained by sorting the entries of $\mathbf{x}$ in descending order.

3.1   Motivation: Graphs and Adjacency Matrices

Matrix representations of graphs in terms of adjacency matrices and Laplacians have been used widely both in applications as well as in the analysis of the structure of graphs based on algebraic properties of these matrices [7]. For example the spectrum of the Laplacian of a graph reveals whether a graph is "diffusive" [24], or whether it is even connected. The degree sequence, which may be obtained from the adjacency matrix or the Laplacian, reveals whether a graph is regular, and it plays a role in a number of real-world investigations of graphs arising in social networks and the Internet.

Given a graph $\mathcal{G}$ defined on n nodes, a *labeling* of the nodes of $\mathcal{G}$ is a function ℓ that maps the nodes of $\mathcal{G}$ onto distinct integers in $\{1, \ldots, n\}$. An adjacency matrix $A \in \mathbf{S}^n$ is then said to *represent* or *specify* $\mathcal{G}$ if there exists a labeling ℓ of the nodes of $\mathcal{G}$ so that the weight of the edge between nodes i and j equals $A_{\ell(i)\ell(j)}$ for all pairs $\{i, j\}$ and the weight of node i equals $A_{\ell(i)\ell(i)}$ for all i. However an adjacency matrix representation A of the graph $\mathcal{G}$ is not unique. In particular $\Pi A \Pi^T$ also specifies $\mathcal{G}$ for all $\Pi \in \mathrm{Sym}(n)$. All these alternative adjacency matrices correspond to different labelings of the nodes of $\mathcal{G}$. *Thus the graph $\mathcal{G}$ is specified by the matrix A only up to a relabeling of the indices of A.* Our objective is to describe abstract structural properties of $\mathcal{G}$ that do *not* depend on a choice of labeling of the nodes. In order to characterize such *unlabeled* graphs in which

the nodes have no distinct identity except through their connections to other nodes, it is important that any function of an adjacency matrix representation of a graph not depend on the particular choice of indices of A. Therefore we seek functions of adjacency matrices that are invariant under conjugation by permutation matrices, and denote such functions as *graph invariants.*

3.2 Definition of Convex Invariants

A convex graph invariant is an invariant that is a convex function of the adjacency matrix of a graph. Specifically we have the following definition:

Definition 3.1. *A function $f : \mathbf{S}^n \to \mathbb{R}$ is a* convex graph invariant *if it is convex, and if for any $A \in \mathbf{S}^n$ it holds that $f(\Pi A \Pi^T) = f(A)$ for all permutation matrices $\Pi \in \text{Sym}(n)$.*

Thus convex graph invariants are convex functions that are *constant over orbits* of the symmetric group acting on symmetric matrices by conjugation. As described above the motivation behind the invariance property is clear. The motivation behind the convexity property is that we wish to construct solutions based on convex programming formulations in order to solve problems such as those listed in Section 2. We present several examples of convex graph invariants in Section 3.3. We note that a *concave graph invariant* is a real-valued function over $\mathbf{S}^n$ that is the negative of a convex graph invariant.

We also consider invariant convex sets, which are defined in an analogous manner to convex graph invariants:

Definition 3.2. *A set $C \subseteq \mathbf{S}^n$ is said to be an* invariant convex set *if it is convex and if for any $A \in C$ it is the case that $\Pi A \Pi^T \in C$ for all permutation matrices $\Pi \in \text{Sym}(n)$.*

In Section 3.4 we present examples in which graphs can be constrained to have various properties by requiring that adjacency matrices belong to such convex invariant sets. We also make connections between robust optimization and invariant convex sets in Section 3.6.

In order to systematically study convex graph invariants, we analyze certain elementary invariants that serve as a basis for constructing arbitrary convex invariants. These elementary invariants are defined as follows:

Definition 3.3. *An* elementary convex graph invariant *is a function $\Theta_P : \mathbf{S}^n \to \mathbb{R}$ of the form*

$$\Theta_P(A) = \max_{\Pi \in \text{Sym}(n)} \text{Tr}(P \Pi A \Pi^T),$$

for any $P \in \mathbf{S}^n$.

It is clear that an elementary invariant is also a convex graph invariant, as it is expressed as the maximum over a set of convex functions (in fact linear functions). We describe various properties of convex graph invariants in Sections 3.5. One useful construction that we give is the expression of arbitrary convex graph invariants as suprema over elementary invariants. We also discuss convex spectral invariants in Section 3.7, which are convex functions of a symmetric matrix that depend purely on its spectrum. Finally an important point is that convex graph invariants may in general be hard to compute. In Section 4 we discuss this problem and propose further tractable convex relaxations for cases in which a convex graph invariant may be intractable to compute.

In the Appendix we describe convex functions defined on $\mathbb{R}^n$ that are invariant with respect to any permutation of the argument. Such functions have been analyzed previously, and we provide a list of their well-known properties. We contrast these properties with those of convex graph invariants throughout the rest of this section.

3.3 Examples of Convex Graph Invariants

We list several examples of convex graph invariants. As mentioned previously some of these invariants may possibly be difficult to compute, but we defer discussion of computational issues to Section 4. A useful property that we exploit in several of these examples is that a function defined as the supremum over a set of convex functions is itself convex [35].

Number of edges. The total number of edges (or sum of edge weights) is an elementary convex graph invariant with $P = \frac{1}{2}(\mathbf{11}^T - I)$.

Node weight. The maximum node weight of a graph, which corresponds to the maximum diagonal entry of the adjacency matrix of the graph, is an elementary convex graph invariant with $P = \mathbf{e}_1\mathbf{e}_1^T$. The maximum diagonal entry in *magnitude* of an adjacency matrix is a convex graph invariant, and can be expressed as follows with $P = \mathbf{e}_1\mathbf{e}_1^T$:

$$\text{max. node weight}(A) = \max\{\Theta_P(A), \Theta_{-P}(A)\}.$$

Similarly the sum of all the node weights, which is the sum of the diagonal entries of an adjacency matrix of a graph, can be expressed as an elementary convex graph invariant with P being the identity matrix.

Maximum degree. The maximum (weighted) degree of a node of a graph is also an elementary convex graph invariant with $P_{1,i} = P_{i,1} = 1$, $\forall i \neq 1$, and all the other entries of P set to zero.

Largest cut. The value of the largest weighted cut of a graph specified by an adjacency matrix $A \in \mathbf{S}^n$ can be written as follows:

$$\text{max. cut}(A) = \max_{\mathbf{y} \in \{-1,+1\}^n} \frac{1}{4} \sum_{i,j} A_{i,j}(1 - \mathbf{y}_i\mathbf{y}_j).$$

As this function is a maximum over a set of linear functions, it is a convex function of A. Further it is also clear that max. $\text{cut}(A) = \text{max. cut}(\Pi A \Pi^T)$ for all permutation matrices Π. Consequently the value of the largest cut of a graph is a convex graph invariant. We note here that computing this invariant is intractable in general. In practice one could instead employ the following well-known tractable SDP relaxation [22], which is related to the MAXCUT value by an appropriate shift and rescaling:

$$
\begin{aligned}
f(A) = \min_{X \in \mathbf{S}^n} \quad & \text{Tr}(XA) \\
\text{s.t.} \quad & X_{ii} = 1, \ \forall i \\
& X \succeq 0.
\end{aligned}
\tag{2}
$$

As this relaxation is expressed as the minimum over a set of linear functions, it is a concave graph invariant. In Section 4.2 we discuss in greater detail tractable relaxations for invariants that are difficult to compute.

Isoperimetric number (Cheeger constant). The isoperimetric number, also known as the Cheeger constant [17], of a graph specified by adjacency matrix $A \in \mathbf{S}^n$ is defined as follows:

$$\text{isoperimetric number}(A) = \min_{U \subset \{1,\ldots,n\}, |U| \leq \frac{n}{2}, \mathbf{y} \in \mathbb{R}^n, \mathbf{y}_U = 1, \mathbf{y}_{U^c} = -1} \sum_{i,j} \frac{A_{i,j}(1 - \mathbf{y}_i\mathbf{y}_j)}{4|U|}.$$

Here $U^c = \{1,\ldots,n\}\backslash U$ denotes the complement of the set U, and $\mathbf{y}_U$ is the subset of the entries of the vector $\mathbf{y}$ indexed by U. As with the last example, it is again clear that this function is a concave graph invariant as it is expressed as the minimum over a set of linear functions. In particular it can be viewed as measuring the value of a "normalized" cut, and plays an important role in several aspects of graph theory [24].

9

Degree sequence invariants. Given a graph specified by adjacency matrix A (assume for simplicity that the node weights are zero), the weighted *degree sequence* is given by the vector $\mathbf{d}(A) = \overline{A\mathbf{1}}$, i.e., the vector obtained by sorting the entries of $A\mathbf{1}$ in descending order. It is easily seen that $\mathbf{d}(A)$ is a graph invariant. Consequently any function of $\mathbf{d}(A)$ is also a graph invariant. However our interest is in obtaining *convex* functions of the adjacency matrix A. An important class of functions of $\mathbf{d}(A)$ that are convex functions of A, and therefore are convex graph invariants, are of the form:

$$f(A) = \mathbf{v}^T \mathbf{d}(A),$$

for $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{v}_1 \geq \cdots \geq \mathbf{v}_n$. This function can also be expressed as the maximum over all permutations $\Pi \in \mathrm{Sym}(n)$ of the inner-product $\mathbf{v}^T \Pi A \mathbf{1}$. As described in the Appendix such linear *monotone* functionals can be used to express *all* convex functions over $\mathbb{R}^n$ that are invariant with respect to permutations of the argument. Consequently these monotone functions serve as building blocks for constructing all convex graph invariants that are functions of $\mathbf{d}(A)$.

Spectral invariants. Let the eigenvalues of the adjacency matrix A of a graph be denoted as $\lambda_1(A) \geq \cdots \geq \lambda_n(A)$, and let $\lambda(A) = [\lambda_1(A), \ldots, \lambda_n(A)]$. These eigenvalues form the *spectrum* of the graph specified by A, and clearly remain unchanged under transformations of the form $A \to VAV^T$ for any orthogonal matrix $V \in \mathrm{O}(n)$ (and therefore for any permutation matrix). Hence any function of the spectrum of a graph is a graph invariant. Analogous to the previous example, an important class of spectral functions that are also *convex* are of the form:

$$f(A) = \mathbf{v}^T \lambda(A),$$

for $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{v}_1 \geq \cdots \geq \mathbf{v}_n$. We denote spectral invariants that are also convex functions as *convex spectral invariants*. As with convex invariants of the degree sequence, all convex spectral invariants can be constructed using monotone functions of the type described here (see the Appendix).

Second-smallest eigenvalue of Laplacian. This example is only meaningful for weighted graphs in which the node and edge weights are non-negative. For such a graph specified by adjacency matrix A, let $D_A = \mathrm{diag}(A\mathbf{1})$, where diag takes as input a vector and forms a diagonal matrix with the entries of the vector on the diagonal. The *Laplacian* of a graph is then defined as follows:

$$L_A = D_A - A.$$

If $A \in \mathbf{S}^n$ consists of nonnegative entries, then $L_A \succeq 0$. In this setting we denote the eigenvalues of L_A as $\lambda_1(L_A) \geq \cdots \geq \lambda_n(L_A)$. It is easily seen that $\lambda_n(L_A) = 0$ as the all-ones vector $\mathbf{1}$ lies in the kernel of L_A. The second-smallest eigenvalue $\lambda_{n-1}(L_A)$ of the Laplacian is a *concave* invariant function of A. It plays an important role as the graph specified by A is connected if and only if $\lambda_{n-1}(L_A) > 0$.

Inverse of Stability Number. A stable set of an unweighted graph $\mathcal{G}$ is a subset of the nodes of $\mathcal{G}$ such that no two nodes in the subset are adjacent. The stability number is the size of the largest stable set of $\mathcal{G}$, and is denoted by $\alpha(\mathcal{G})$. By a result of Motzkin and Straus [32], the inverse of the stability number can be written as follows:

$$
\begin{aligned}
\frac{1}{\alpha(\mathcal{G})} = \min_{\mathbf{x}} \quad & \mathbf{x}^T(I + A)\mathbf{x} \\
\text{s.t.} \quad & \mathbf{x}_i \geq 0, \ \forall i, \quad \sum_i \mathbf{x}_i = 1.
\end{aligned}
\tag{3}
$$

Here A is any adjacency matrix representing the graph $\mathcal{G}$. Although this formulation is for unweighted graphs with edge weights being either one or zero, we note that the definition can in fact

be *extended* to all weighted graphs, i.e., for graphs with adjacency matrix given by *any* $A \in \mathbf{S}^n$. Consequently, the inverse of this extended stability number of a graph is a concave graph invariant over $\mathbf{S}^n$ as it is expressed as the minimum over a set of linear functions. As this function is difficult to compute in general (because the stability number of a graph is intractable to compute), one could employ the following tractable relaxation:

$$f(A) = \min_{X \in \mathbf{S}^n} \quad \mathrm{Tr}(X(I + A))$$
$$\text{s.t.} \quad X \geq 0, \quad X \succeq 0, \quad \mathbf{1}^T X \mathbf{1} = 1. \tag{4}$$

This relaxation is also a concave graph invariant as it is expressed as the minimum over a set of affine functions.

3.4 Examples of Invariant Convex Sets

Next we provide examples of invariant convex sets. As described below constraints expressed using such sets are useful in order to require that graphs have certain properties. Note that a sublevel set $\{A : f(A) \leq \alpha\}$ for any convex graph invariant f is an invariant convex set. Therefore all the examples of convex graph invariants given above can be used to construct invariant convex set constraints.

Algebraic connectivity and diffusion. As mentioned in Section 3.3 a graph represented by adjacency matrix $A \in \mathcal{A}$ has the property that the second-smallest eigenvalue $\lambda_{n-1}(L_A)$ of the Laplacian of the graph is a concave graph invariant. The constraint set $\{A : A \in \mathcal{A}, \lambda_{n-1}(L_A) \geq \epsilon\}$ for any $\epsilon > 0$ expresses the property that a graph must be *connected*. Further if we set ϵ to be relatively large, we can require that a graph has good diffusion properties.

Largest clique constraint. Let $K_k \in \mathbf{S}^n$ denote the adjacency matrix of an unweighted k-clique. Note that K_k is only nonzero within a $k \times k$ submatrix, and is zero-padded to lie in $\mathbf{S}^n$. Consider the following invariant convex set for $\epsilon > 0$:

$$\{A : A \in \mathcal{A}, \ \Theta_{K_k}(A) \leq (k^2 - k) - \epsilon\}.$$

This constraint set expresses the property that a graph cannot have a clique of size k (or larger), with the edge weights of all edges in the clique being close to one. For example we can use this constraint set to require that a graph has no triangles (with large edge weights). It is important to note that triangles (and cliques more generally) are forbidden only with the qualification that all the edge weights in the triangle cannot be close to one. For example a graph may contain a triangle with each edge having weight equal to $\frac{1}{2}$. In this case the function Θ_{K_3} evaluates to 3, which is much smaller than the maximum value of 6 that Θ_{K_3} can take for matrices in $\mathcal{A}$ that contain a triangle with edge weights equal to one.

Girth constraint. The girth of a graph is the length of the shortest cycle. Let $C_k \in \mathbf{S}^n$ denote the adjacency matrix of an unweighted k-cycle for $k \leq n$. As with the k-clique note that C_k is nonzero only within a $k \times k$ submatrix, and is zero-padded so that it lies in $\mathbf{S}^n$. In order to express the property that a graph has no small cycles, consider the following invariant convex set for $\epsilon > 0$:

$$\{A : A \in \mathcal{A}, \ \Theta_{C_k}(A) \leq 2k - \epsilon \ \forall k \leq k_0\}.$$

Graphs belonging to this set cannot have cycles of length less than or equal to k_0, with the weights of edges in the cycle being close to one. Thus we can impose a lower bound on a weighted version of the girth of a graph.

Forbidden subgraph constraint. The previous two examples can be viewed as special cases of a more general constraint involving forbidden subgraphs. Specifically let A_k denote the adjacency

11

matrix of an unweighted graph on k nodes that consists of E_k edges. As before A_k is zero-padded to ensure that it lies in $\mathbf{S}^n$. Consider the following invariant convex set for $\epsilon > 0$:

$$\{A : A \in \mathcal{A}, \; \Theta_{A_k}(A) \leq 2E_k - \epsilon\}.$$

This constraint set requires that a graph not contain the subgraph given by the adjacency matrix A_k, with edge weights close to one.

Degree distribution. Using the notation described previously, let $\mathbf{d}(A) = \overline{A\mathbf{1}}$ denote the sorted degree sequence $(\mathbf{d}(A)_1 \geq \cdots \geq \mathbf{d}(A)_n)$ of a graph specified by adjacency matrix A. We wish to consider the set of all graphs that have degree sequence $\mathbf{d}(A)$. This set is in general not convex unless A represents a (weighted) regular graph, i.e., $\mathbf{d}(A) = \alpha\mathbf{1}$ for some constant α. Therefore we consider the *convex hull* of all graphs that have degree sequence given by $\mathbf{d}$:

$$\mathcal{D}(A) = \operatorname{conv}\{B : B \in \mathbf{S}^n, \; \overline{B\mathbf{1}} = \mathbf{d}(A)\}.$$

This set is in fact tractable to represent, and is given by the set of graphs whose degree sequence is *majorized* by $\mathbf{d}$:

$$\mathcal{D}(A) = \left\{B : B \in \mathbf{S}^n, \; \mathbf{1}^T B\mathbf{1} = \mathbf{1}^T\mathbf{d}(A), \; \sum_{i=1}^{k}(\overline{B\mathbf{1}})_i \leq \sum_{i=1}^{k}\mathbf{d}(A)_i \; \forall k = 1, \ldots, n-1\right\}.$$

By the majorization principle [5] another representation for this convex set is as the set of graphs whose degree sequence lies in the *permutahedron* generated by $\mathbf{d}$ [41]; the permutahedron generated by a vector is the convex hull of all permutations of the vector. The notion of majorization is sometimes also referred to as *Lorenz dominance* (see the Appendix for more details).

Spectral distribution. Let $\lambda(A)$ denote the spectrum of a graph represented by adjacency matrix A. As before we are interested in the set of all graphs that have spectrum $\lambda(A)$. This set is nonconvex in general, unless A is a multiple of the identity matrix in which case all the eigenvalues are the same. Therefore we consider the convex hull of all graphs (i.e., symmetric adjacency matrices) that have spectrum equal to $\lambda(A)$:

$$\mathcal{E}(A) = \operatorname{conv}\{B : B \in \mathbf{S}^n, \; \lambda(B) = \lambda(A)\}.$$

This convex hull also has a tractable semidefinite representation analogous to the description above [5]:

$$\mathcal{E}(A) = \left\{B : B \in \mathbf{S}^n, \; \operatorname{Tr}(B) = \operatorname{Tr}(A), \; \sum_{i=1}^{k}\lambda(B)_i \leq \sum_{i=1}^{k}\lambda(A)_i \; \forall k = 1, \ldots, n-1\right\}.$$

Note that eigenvalues are specified in descending order, so that $\sum_{i=1}^{k}\lambda(B)_i$ represents the sum of the k-largest eigenvalues of B.

3.5 Representation of Convex Graph Invariants

All invariant convex sets and convex graph invariants can be represented using elementary convex graph invariants. In this section we describe both these representation results. Representation theorems in mathematics give expressions of complicated sets or functions in terms of simpler, basic objects. In functional analysis the Riesz representation theorem relates elements in a Hilbert space and its dual, by uniquely associating each element of the Hilbert space to a linear functional [37]. In probability theory de Finetti's theorem states that a collection of exchangeable random

12

variables can be expressed as a mixture of independent, identically distributed random variables. In convex analysis every closed convex set can be expressed as the intersection of halfspaces [35]. In each of these cases representation theorems provide a powerful analysis tool as they give a *canonical* expression for complicated mathematical objects in terms of elementary sets/functions.

First we give a representation result for convex graph invariants. In order to get a flavor of this result consider the maximum absolute-value node weight invariant of Section 3.3, which is represented as the supremum over two elementary convex graph invariants. The following theorem states that in fact any convex graph invariant can be expressed as a supremum over elementary invariants:

Theorem 3.4. *Let f be any convex graph invariant. Then f can be expressed as follows:*

$$f(A) = \sup_{P \in \mathcal{P}} \Theta_P(A) - \alpha_P,$$

for $\alpha_P \in \mathbb{R}$ and for some subset $\mathcal{P} \subset \mathbf{S}^n$.

Proof. Since f is a convex function, it can be expressed as the supremum over linear functionals as follows:

$$f(A) = \sup_{P \in \mathcal{P} \subseteq \mathbf{S}^n} \mathrm{Tr}(PA) - \alpha_P,$$

for $\alpha_P \in \mathbb{R}$. This conclusion follows directly from the separation theorem in convex analysis [35]; in particular this description of the convex function f can be viewed as a specification in terms of supporting hyperplanes of the epigraph of f, which is a convex subset of $\mathbf{S}^n \times \mathbb{R}$. However as f is also a graph invariant, we have that $f(A) = f(\Pi A \Pi^T)$ for any permutation Π and for all $A \in \mathbf{S}^n$. Consequently for any permutation Π and for any $P \in \mathcal{P}$,

$$f(A) = f(\Pi A \Pi^T) \geq \mathrm{Tr}(P \Pi A \Pi^T) - \alpha_P.$$

Thus we have that

$$f(A) \geq \sup_{P \in \mathcal{P}} \Theta_P(A) - \alpha_P. \tag{5}$$

However it also clear that for each $P \in \mathcal{P}$

$$\Theta_P(A) - \alpha_P \geq \mathrm{Tr}(PA) - \alpha_P,$$

which allows us to conclude that

$$\sup_{P \in \mathcal{P}} \Theta_P(A) - \alpha_P \geq \sup_{P \in \mathcal{P}} \mathrm{Tr}(PA) - \alpha_P = f(A). \tag{6}$$

Combining equations (5) and (6) we have the desired result. $\square$

Remark 3.5. *This result can be strengthened in the sense that one need only consider elements in $\mathcal{P}$ that lie in different equivalence classes up to conjugation by permutation matrices $\Pi \in \mathrm{Sym}(n)$. In each equivalence class the representative functional is the one with the smallest value of α_P. This idea can be formalized as follows. Consider the group action $\rho : (M, \Pi) \rightarrowtail \Pi M \Pi^T$ that conjugates elements in $\mathbf{S}^n$ by a permutation matrix in $\mathrm{Sym}(n)$. With this notation we may restrict our attention in Theorem 3.4 to $\mathcal{P} \subset \mathbf{S}^n/\mathrm{Sym}(n)$, where $\mathbf{S}^n/\mathrm{Sym}(n)$ represents the quotient space under the group action ρ. Such a mathematical object obtained by taking the quotient of a Euclidean space (or more generally a smooth manifold) under the action of a finite group is called an* orbifold. *With this strengthening one can show that there exists a* unique, minimal *representation set $\mathcal{P} \subset \mathbf{S}^n/\mathrm{Sym}(n)$. We however do not emphasize such refinements in subsequent results, and stick with the weaker statement that $\mathcal{P} \subseteq \mathbf{S}^n$ for notational and conceptual simplicity.*

As our next result we show that any invariant convex set can be represented as the intersection of sublevel sets of elementary convex graph invariants:

Proposition 3.6. *Let $\mathcal{S} \subseteq \mathbf{S}^n$ be an invariant convex set. Then there exists a representation of $\mathcal{S}$ as follows:*

$$\mathcal{S} = \bigcap_{P \in \mathcal{P}} \{A : A \in \mathbf{S}^n, \ \Theta_P(A) \leq \alpha_P\},$$

for some $\mathcal{P} \subseteq \mathbf{S}^n$ and for $\alpha_P \in \mathbb{R}$.

Proof. The proof of this statement proceeds in an analogous manner to that of Theorem 3.4, and is again essentially a consequence of the separation theorem in convex analysis. $\qquad\square$

3.6 A Robust Optimization View of Invariant Convex Sets

Uncertainty arises in many real-world problems. An important goal in robust optimization (see [4] and the reference therein) is to translate formal notions of measures of uncertainty into convex constraint sets. Convexity is important in order to obtain optimization formulations that are tractable.

The representation of a graph via an adjacency matrix in $\mathbf{S}^n$ is inherently uncertain as we have no information about the specific labeling of the nodes of the graph. In this section we associate to each graph a convex polytope, which represents the best convex uncertainty set given a graph:

Definition 3.7. *Let $\mathcal{G}$ be a graph that is represented by an adjacency matrix $A \in \mathbf{S}^n$ (any choice of representation is suitable). The* convex hull *of the graph $\mathcal{G}$ is defined as the following convex polytope:*

$$\mathcal{C}(\mathcal{G}) = \text{conv}\{\Pi A \Pi^T : \Pi \in \text{Sym}(n)\}.$$

Recall that $\text{Sym}(n)$ is the symmetric group of $n \times n$ permutation matrices. One can check that the convex hull of a graph is an invariant convex set, and that its extreme points are the matrices $\Pi A \Pi^T$ for all $\Pi \in \text{Sym}(n)$. Note that this convex hull may in general be intractable to characterize; if these polytopes were tractable to characterize we would be able to solve the graph isomorphism problem in polynomial time.

The convex hull of a graph is the smallest convex set that contains all the adjacency matrices that represent the graph. Therefore $\mathcal{C}(\mathcal{G})$ is in some sense the "best convex characterization" of the graph $\mathcal{G}$. This notion is related to the concept of *risk measures* studied in [3], and the construction of convex uncertainty sets based on these risk measures studied in [6]. In particular we recall the following definition from [6]:

Definition 3.8. *Let $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$ be any finite collection of elements with $Z_i \in \mathbf{S}^n$. Let $\mathbf{q} \in \mathbb{R}^k$ be a probability distribution, i.e., $\sum_i \mathbf{q}_i = 1$ and $\mathbf{q}_i \geq 0$, $\forall i$. Then the $\mathbf{q}$-permutohull is the polytope in $\mathbf{S}^n$ defined as follows:*

$$\mathcal{B}_{\mathbf{q}}(\mathcal{Z}) = \text{conv}\left\{\sum_i (\Pi \mathbf{q})_i Z_i : \Pi \in \text{Sym}(k)\right\}.$$

Convex uncertainty sets given by permutohulls emphasize a data-driven view of robust optimization as adopted in [6]. Specifically the only information available about an uncertain set in many settings is a finite collection of data vectors $\mathcal{Z}$, and the probability distribution $\mathbf{q}$ expresses preferences over such an unordered data set. Therefore given a data set and a probability distribution that quantifies uncertainty with respect to elements of this data set, the $\mathbf{q}$-permutohull is the

smallest convex set expressing these uncertainty preferences. We note that an important property of a permutohull is that it is invariant with respect to relabeling of the data vectors in $\mathcal{Z}$.

The convex hull of a graph $\mathcal{C}(\mathcal{G})$ is a simple example of a permutohull $\mathcal{B}_{\mathbf{q}}(\mathcal{Z})$, with the distribution being $\mathbf{q} = (1, 0, \ldots, 0)$ and the set $\mathcal{Z} = \{\Pi A \Pi^T : \Pi \in \mathrm{Sym}(n)\}$ where $A \in \mathbf{S}^n$ represents the graph $\mathcal{G}$. More complicated permutohulls of graphs may be of interest in several applications but we do not pursue these generalizations here, and instead focus on the case of the convex hull of a graph as defined above.

The convex hull of a graph is itself an invariant convex set by definition. Therefore we can appeal to Proposition 3.6 to give a representation of this set in terms of sublevel sets of elementary convex graph invariants. As our next result we show that the values of all elementary convex graph invariants of $\mathcal{G}$ can be used to produce such a representation:

Proposition 3.9. *Let $\mathcal{G}$ be a graph and let $A \in \mathbf{S}^n$ be an adjacency matrix representing $\mathcal{G}$. We then have that*
$$\mathcal{C}(\mathcal{G}) = \bigcap_{P \in \mathbf{S}^n} \{B : B \in \mathbf{S}^n, \ \Theta_P(B) \leq \Theta_P(A)\}.$$

Proof. One direction of inclusion in this result is easily seen. Indeed we have that for any $\Pi \in \mathrm{Sym}(n)$
$$\Pi A \Pi^T \in \bigcap_{P \in \mathbf{S}^n} \{B : B \in \mathbf{S}^n, \ \Theta_P(B) \leq \Theta_P(A)\}.$$

As the right-hand-side is a convex set it is clear that the convex hull $\mathcal{C}(\mathcal{G})$ belongs to the set on the right-hand-side:
$$\mathcal{C}(\mathcal{G}) \subseteq \bigcap_{P \in \mathbf{S}^n} \{B : B \in \mathbf{S}^n, \ \Theta_P(B) \leq \Theta_P(A)\}.$$

For the other direction suppose for the sake of a contradiction that we have a point $M \notin \mathcal{C}(\mathcal{G})$ but with $\Theta_P(M) \leq \Theta_P(A)$ for all $P \in \mathbf{S}^n$. As $M \notin \mathcal{C}(\mathcal{G})$ we appeal to the separation theorem from convex analysis [35] to produce a strict separating hyperplane between M and $\mathcal{C}(\mathcal{G})$, i.e., a $\tilde{P} \in \mathbf{S}^n$ such that
$$\mathrm{Tr}(\tilde{P}B) < \alpha, \forall B \in \mathcal{C}(\mathcal{G}), \quad \text{and} \quad \mathrm{Tr}(\tilde{P}M) > \alpha.$$

Further as $\mathcal{C}(\mathcal{G})$ is an invariant convex set, it must be the case that
$$\Theta_{\tilde{P}}(B) < \alpha, \forall B \in \mathcal{C}(\mathcal{G}).$$

On the other hand as $\mathrm{Tr}(\tilde{P}M) > \alpha$ we also have that $\Theta_{\tilde{P}}(M) > \alpha$. It is thus clear that
$$\Theta_{\tilde{P}}(A) < \alpha < \Theta_{\tilde{P}}(M),$$

which leads us to a contradiction and concludes the proof. □

Therefore elementary convex graph invariants are useful for representing all the "convex properties" of a graph. This result agrees with the intuition that the "maximum amount of information" that one can hope to obtain from convex graph invariants about a graph should be limited fundamentally by the convex hull of the graph.

As mentioned previously in many cases the convex hull of a graph may be intractable to characterize. One can obtain outer bounds to this convex hull by using a tractable subset of elementary convex graph invariants; therefore we may obtain tractable but weaker convex uncertainty sets than the convex hull of a graph. From Proposition 3.9 such approximations can be refined as we use additional elementary convex graph invariants. As an example the spectral convex constraint sets described in Section 3.4 provide a tractable relaxation that plays a prominent role in our experiments in Section 4.

15

3.7 Comparison with Spectral Invariants

Convex functions that are invariant under certain group actions have been studied previously. The most prominent among these is the set of convex functions of symmetric matrices that are invariant under conjugation by orthogonal matrices [13]:

$$f(M) = f(VMV^T), \ \forall \ M \in \mathbf{S}^n, \ \forall \ V \in \mathrm{O}(n).$$

It is clear that such functions depend only on the spectrum of a symmetric matrix, and therefore we refer to them as *convex spectral invariants*:

$$f(M) = \tilde{f}(\lambda(M)),$$

where $\tilde{f} : \mathbb{R}^n \to \mathbb{R}$. It is shown in [13] that f is convex if and only if $\tilde{f}$ is a convex function that is *symmetric* in its argument:

$$\tilde{f}(\mathbf{x}) = \tilde{f}(\Pi\mathbf{x}), \ \forall \mathbf{x} \in \mathbb{R}^n, \forall \Pi \in \mathrm{Sym}(n).$$

One can check that any convex spectral invariant can be represented as the supremum over monotone functionals of the spectrum of the form:

$$\tilde{f}(\mathbf{x}) = \mathbf{v}^T \overline{\mathbf{x}} - \alpha,$$

for $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{v}_1 \geq \cdots \geq \mathbf{v}_n$. See the Appendix for more details.

A convex spectral invariant is also a convex graph invariant as invariance with respect to conjugation by any orthogonal matrix is a stronger requirement than invariance with respect to conjugation by any permutation matrix. As many convex spectral invariants are tractable to compute, they form an important subclass of convex graph invariants. In Section 4.1 we discuss a natural approximation to elementary convex graph invariants using convex spectral invariants by replacing the symmetric group $\mathrm{Sym}(n)$ in the maximization by the orthogonal group $\mathrm{O}(n)$. Finally one can define a spectrally invariant convex set $\mathcal{S}$ (analogous to invariant convex sets defined in Section 3.2) in which $M \in \mathcal{S}$ if and only if $VMV^T \in \mathcal{S}$ for all $V \in \mathrm{O}(n)$. Such sets are very useful in order to impose various spectral constraints on graphs, and often have tractable semidefinite representations.

3.8 Convex versus Non-Convex Invariants

There are many graph invariants that are not convex. In this section we give two examples that serve to illustrate the strengths and weaknesses of convex graph invariants. First consider the spectral invariant given by the fifth largest eigenvalue of a graph, i.e., $\lambda_5(A)$ for a graph specified by adjacency matrix A. This function is a graph invariant but it is not convex. However from Section 3.3 we have that the *sum* of the first five eigenvalues of a graph is a convex graph invariant. More generally any function of the form $v_1\lambda_1 + \cdots + v_5\lambda_5$ with $v_1 \geq \cdots \geq v_5$ is a convex graph invariant. Thus information about the fifth eigenvalue can be obtained in a "convex manner" only by including information about all the top five eigenvalues (or all the bottom $n - 4$ eigenvalues). As a second example consider the (weighted) sum of the total number of triangles that occur as subgraphs in a graph. This function is again a non-convex graph invariant. However recall from the forbidden subgraph example in Section 3.4 that we can use elementary convex graph invariants to test whether a graph contains a triangle as a subgraph (with the edges of the triangle having large weights). Therefore, roughly speaking convex graph invariants can be used to decide whether a

graph contains a triangle, while general non-convex graph invariants can provide more information about the total number of triangles in a graph. These examples demonstrate that convex graph invariants have certain limitations in terms of the type of information that they can convey about a graph.

The weaker form of information about a graph conveyed by convex graph invariants is nonetheless still useful in distinguishing between graphs. As the next result demonstrates convex graph invariants are strong enough to distinguish between non-isomorphic graphs. This lemma follows from a straightforward application of Proposition 3.9:

Lemma 3.10. *Let $\mathcal{G}_1, \mathcal{G}_2$ be two non-isomorphic graphs represented by adjacency matrices $A_1, A_2 \in \mathbf{S}^n$, i.e., there exists no permutation $\Pi \in \mathrm{Sym}(n)$ such that $A_1 = \Pi A_2 \Pi^T$. Then there exists a $P \in \mathbf{S}^n$ such that $\Theta_P(A_1) \neq \Theta_P(A_2)$.*

Proof. Assume for the sake of a contradiction that $\Theta_P(A_1) = \Theta_P(A_2)$ for all $P \in \mathbf{S}^n$. Then we have from Proposition 3.9 that $\mathcal{C}(\mathcal{G}_1) = \mathcal{C}(\mathcal{G}_2)$. As the extreme points of these polytopes must be the same, there must exist a permutation $\Pi \in \mathrm{Sym}(n)$ such that $A_1 = \Pi A_2 \Pi^T$. This leads to a contradiction. $\square$

Hence for any two given non-isomorphic graphs there exists an elementary convex graph invariant that evaluates to different values for these two graphs. Consequently elementary convex graph invariants form a *complete* set of graph invariants as they can distinguish between any two non-isomorphic graphs.

4 Computing Convex Graph Invariants

In this section we focus on efficiently computing and approximating convex graph invariants, and on tractable representations of invariant convex sets. We begin by studying the question of computing elementary convex graph invariants, before moving on to more general convex invariants.

4.1 Elementary Invariants and the Quadratic Assignment Problem

As all convex graph invariants can be represented using only elementary invariants, we initially focus on computing the latter. Computing an elementary convex graph invariant $\Theta_P(A)$ for general A, P is equivalent to solving the so-called Quadratic Assignment Problem (QAP) [11]. Solving QAP is hard in general, because it includes as a special case the Hamiltonian cycle problem; if P is the adjacency matrix of the n-cycle, then for an unweighted graph specified by adjacency matrix A we have that $\Theta_P(A)$ is equal to $2n$ if and only if the graph contains a Hamiltonian cycle. However there are well-studied spectral and semidefinite relaxations for QAP, which we discuss next.

The *spectral relaxation* of $\Theta_P(A)$ is obtained by replacing the symmetric group $\mathrm{Sym}(n)$ in the definition by the orthogonal group $\mathrm{O}(n)$:

$$\Lambda_P(A) = \max_{V \in \mathrm{O}(n)} \mathrm{Tr}(PVAV^T). \tag{7}$$

Clearly $\Theta_P(A) \leq \Lambda_P(A)$ for all $A, P \in \mathbf{S}^n$. As one might expect $\Lambda_P(A)$ has a simple closed-form solution [19]:

$$\Lambda_P(A) = \lambda(P)^T \lambda(A), \tag{8}$$

where $\lambda(A), \lambda(P)$ are the eigenvalues of A, P sorted in descending order.

The spectral relaxation offers a simple bound, but is quite weak in many instances. Next we consider the well-studied *semidefinite relaxation* for the QAP, which offers a tighter relaxation [40]. The main idea behind the semidefinite relaxation is that we can linearize $\Theta_P(A)$ as follows:

$$
\begin{aligned}
\Theta_P(A) &= \max_{\Pi \in \mathrm{Sym}(n)} \mathrm{Tr}(P\Pi A\Pi^T) \\
&= \max_{\mathbf{x} \in \mathbb{R}^{n^2}, \mathbf{x}=\mathrm{vec}(\Pi), \Pi \in \mathrm{Sym}(n)} \langle \mathbf{x}, (A \otimes P)\mathbf{x} \rangle \\
&= \max_{\mathbf{x} \in \mathbb{R}^{n^2}, \mathbf{x}=\mathrm{vec}(\Pi), \Pi \in \mathrm{Sym}(n)} \mathrm{Tr}((A \otimes P)\mathbf{x}\mathbf{x}^T).
\end{aligned}
$$

Here $A \otimes P$ denotes the tensor product between A and P, and vec denotes the operation that stacks the columns of a matrix into a single vector. Consequently it is of interest to characterize the following convex hull:

$$
\mathrm{conv}\{\mathbf{x}\mathbf{x}^T : \mathbf{x} \in \mathbb{R}^{n^2},\ \mathbf{x} = \mathrm{vec}(\Pi),\ \Pi \in \mathrm{Sym}(n)\}.
$$

There is no known tractable characterization of this set, and by considering tractable approximations the semidefinite relaxation to $\Theta_P(A)$ is then obtained as follows:

$$
\begin{aligned}
\Omega_P(A) = \max_{\mathbf{y} \in \mathbb{R}^{n^2},\ Y \in \mathbf{S}(n^2)} \quad & \mathrm{Tr}(P \otimes A) \\
\text{s.t.} \quad & \mathrm{Tr}((I \otimes (J - I))Y + ((J - I) \otimes I)Y) = 0 \\
& \mathrm{Tr}(Y) - 2\mathbf{y}^T \mathbf{1} = -n \\
& Y \geq 0,\ \begin{pmatrix} 1 & \mathbf{y}^T \\ \mathbf{y} & Y \end{pmatrix} \succeq 0.
\end{aligned}
\tag{9}
$$

We refer the reader to [40] for the detailed steps involved in the construction of this relaxation. This SDP relaxation gives an upper bound to $\Theta_P(A)$, i.e., $\Omega_P(A) \geq \Theta_P(A)$. One can show that if the extra rank constraint

$$
\mathrm{rank}\begin{pmatrix} 1 & \mathbf{y}^T \\ \mathbf{y} & Y \end{pmatrix} = 1
$$

is added to the SDP (9), then $\Omega_P(A) = \Theta_P(A)$. Therefore if the optimal value of the SDP (9) is achieved at some $\hat{\mathbf{y}}, \hat{Y}$ such that this rank-one constraint is satisfied, then the relaxation is tight, i.e., we would have that $\Omega_P(A) = \Theta_P(A)$.

While the semidefinite relaxation (9) can in principle be computed in polynomial-time, the size of the variable $Y \in \mathbf{S}(n^2)$ means that even moderate size problem instances are not well-suited to solution by interior-point methods. In many practical situations however, we often have that the matrix $P \in \mathbf{S}^n$ represents the adjacency matrix of some small graph on k nodes with $k \ll n$, i.e., P is nonzero only inside a $k \times k$ submatrix and is zero-padded elsewhere so that it lies in $\mathbf{S}^n$. For example as discussed in Section 3.4, P may represent the adjacency matrix of a triangle in a constraint expressing that a graph is triangle-free. In such cases computing or approximating $\Theta_P(A)$ may be done more efficiently as follows:

1. **Combinatorial enumeration.** For very small values of k it is possible to compute $\Theta_P(A)$ efficiently even by explicit combinatorial enumeration. The complexity of such a procedure scales as $\mathcal{O}(n^k)$. This approach may be suitable if, for example, P represents the adjacency matrix of a triangle.

2. **Symmetry reduction.** For larger values of k, combinatorial enumeration may no longer be appropriate. In these cases the special structure in P can be exploited to reduce the size of the SDP relaxation (9). Specifically, using the methods described in [14] it is possible to reduce the size of the matrix variables from $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ to size $\mathcal{O}(kn) \times \mathcal{O}(kn)$. More generally, it is also possible to exploit *group symmetry* in P to similarly reduce the size of the SDP (9) (see [14] for details).

4.2 Other Methods and Computational Issues

In many special cases in which computing convex graph invariants may be intractable, it is also possible to use other types of tractable semidefinite relaxations. As described in Section 3.3 the MAXCUT value and the inverse stability number of graphs are invariants that are respectively convex and concave. However both of these are intractable to compute, and as a result we must employ the SDP relaxations for these invariants as discussed in Section 3.3.

Another issue that arises in practice is the *representation* of invariant convex sets. As an example, let $f(A)$ denote the SDP relaxation of the MAXCUT value as defined in (2). As $f(A)$ is a concave graph invariant, we may be interested in representing convex constraint sets as follows:

$$\{A : A \in \mathbf{S}^n, \ f(A) \geq \alpha\} = \{A : A \in \mathbf{S}^n, \ \mathrm{Tr}(XA) \geq \alpha \ \forall X \in \mathbf{S}^n \text{ s.t. } X_{ii} = 1, \ X \succeq 0\}.$$

In order to computationally represent such a set specified in terms of a universal quantifier, we appeal to convex duality. Using the standard dual formulation of (2), we have that:

$$\{A : A \in \mathbf{S}^n, \ f(A) \geq \alpha\} = \{A : A \in \mathbf{S}^n, \ \exists Y \text{ diagonal s.t. } A \succeq Y, \ \mathrm{Tr}(Y) \geq \alpha\}.$$

This reformulation provides a description in terms of existential quantifiers that is more suitable for practical representation. Such reformulations using convex duality are well-known, and can be employed more generally (e.g., for invariant convex sets specified by sublevel sets of the inverse stability number or its relaxations in Section 3.3)

5 Using Convex Graph Invariants in Applications

In this section we give solutions to the stylized problems of Section 2 using convex graph invariants. In order to properly state our results we begin with a few definitions. All the convex programs in our numerical experiments are solved using a combination of the SDPT3 package [38] and the YALMIP parser [29].

5.1 Preliminary Definitions

Let C be a convex set in $\mathbf{S}^n$, and let $\mathbf{x} \in C$ be any point in C. Following standard notions from convex analysis [35], the *tangent cone* at $\mathbf{x}$ with respect to C is defined as follows:

Definition 5.1. *Given a convex set C, the* tangent cone *at a point $\mathbf{x} \in C$ with respect to C is the set of directions from $\mathbf{x}$ to any other point in C:*

$$\mathcal{T}_C(\mathbf{x}) = \{\alpha\mathbf{z} : \mathbf{z} = \mathbf{y} - \mathbf{x}, \mathbf{y} \in C, \alpha \geq 0\}.$$

If C is a convex set expressing a constraint in a convex program, the tangent cone at a point $\mathbf{x} \in C$ can be viewed as the set of feasible directions at $\mathbf{x}$ to other points in C. Next we define the *normal cone* at $\mathbf{x}$ with respect to C, again following the usual conventions in convex analysis [35]:

Definition 5.2. *Given a convex set C, the* normal cone *at a point $\mathbf{x} \in C$ with respect to C is the set of normal vectors to supporting hyperplanes of C at $\mathbf{x}$:*

$$\mathcal{N}_C(\mathbf{x}) = \{\mathbf{z} : \langle \mathbf{z}, \mathbf{y} - \mathbf{x} \rangle \leq 0 \; \forall \mathbf{y} \in C\}.$$

The normal cone and the tangent cone are polars of each other [35]. A key property of normal cones that we use in stating our results is that for any convex set $C \subseteq \mathbf{S}^n$, the normal cones at all the extreme points of C form a *partition*[1] of $\mathbf{S}^n$ [35].

5.2 Application: Graph Deconvolution

Given a combination of two graphs overlaid on the same set of nodes, the graph deconvolution problem is to recover the individual graphs (as introduced in Section 2.1).

Problem 1. *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two graphs specified by particular adjacency matrices $A_1^*, A_2^* \in \mathbf{S}^n$. We are given the sum $A = A_1^* + A_2^*$, and the additional information that A_1^*, A_2^* correspond to particular realizations (labelings of nodes) of $\mathcal{G}_1, \mathcal{G}_2$. The goal is to recover A_1^* and A_2^* from A.*

See Figure 1 for an example illustrating this problem. The key unknown in this problem is the specific labeling of the nodes of $\mathcal{G}_1$ and $\mathcal{G}_2$ relative to each other in the composite graph represented by A. As described in Section 3.6, the best convex constraints that express this uncertainty are the convex hulls of the graphs $\mathcal{G}_1, \mathcal{G}_2$. Therefore we consider the following natural solution based on convex optimization to solve the deconvolution problem:

Solution 1. *Recall that $\mathcal{C}(\mathcal{G}_1)$ and $\mathcal{C}(\mathcal{G}_2)$ are the convex hulls of the unlabeled graphs $\mathcal{G}_1, \mathcal{G}_2$ (which we are given), and that $\|\cdot\|$ denotes the Euclidean norm. We propose the following convex program to recover A_1, A_2:*

$$
(\hat{A}_1, \hat{A}_2) = \arg \min_{A_1, A_2 \in \mathbf{S}^n} \quad \|A - A_1 - A_2\| \tag{10}
$$
$$
\text{s.t.} \quad A_1 \in \mathcal{C}(\mathcal{G}_1), \; A_2 \in \mathcal{C}(\mathcal{G}_2).
$$

One could also use in the objective any other norm that is invariant under conjugation by permutation matrices. This program is convex, although it may not be tractable if the sets $\mathcal{C}(\mathcal{G}_1), \mathcal{C}(\mathcal{G}_2)$ cannot be efficiently represented. Therefore it may be desirable to use tractable convex relaxations C_1, C_2 of the sets $\mathcal{C}(\mathcal{G}_1), \mathcal{C}(\mathcal{G}_2)$, i.e., $\mathcal{C}(\mathcal{G}_1) \subseteq C_1 \subset \mathbf{S}^n$ and $\mathcal{C}(\mathcal{G}_2) \subseteq C_2 \subset \mathbf{S}^n$:

$$
(\hat{A}_1, \hat{A}_2) = \arg \min_{A_1, A_2 \in \mathbf{S}^n} \quad \|A - A_1 - A_2\| \tag{11}
$$
$$
\text{s.t.} \quad A_1 \in C_1, \; A_2 \in C_2.
$$

Recall from Proposition 3.9 that we can represent $\mathcal{C}(\mathcal{G})$ using all the elementary convex graph invariants. Tractable relaxations to this convex hull may be obtained, for example, by just using spectral invariants, degree-sequence invariants, or any other subset of invariant convex set constraints that can be expressed efficiently. We give numerical examples later in this section. The following result gives conditions under which we can exactly recover A_1^*, A_2^* using the convex program (11):

Proposition 5.3. *Given the problem setup as described above, we have that $(\hat{A}_1, \hat{A}_2) = (A_1^*, A_2^*)$ is the* unique optimum *of (11) if and only if:*

$$T_{C_1}(A_1^*) \cap -T_{C_2}(A_2^*) = \{0\},$$

where $-T_{C_2}(A_2^*)$ denotes the negative of the tangent cone $T_{C_2}(A_2^*)$.

---

[1]Note that there may be overlap on the boundaries of the normal cones at the extreme points, but these overlaps have smaller dimension than those of the normal cones.
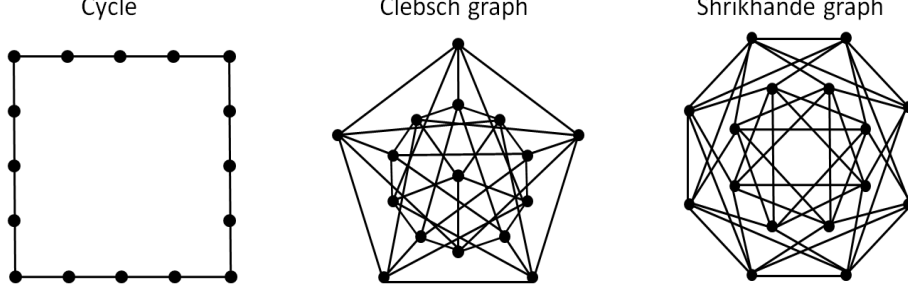
Figure 3: The three graphs used in the deconvolution experiments of Section 5.2. The Clebsch graph and the Shrikhande graph are examples of strongly regular graphs on 16 nodes [21]; see Section 5.2 for more details about the properties of such graphs.

Proof. Note that in the setup described above (A_1^*, A_2^*) is an optimal solution of the convex program (11) as this point is feasible (since by construction $A_1^* \in \mathcal{C}(\mathcal{G}_1) \subseteq C_1$ and $A_2^* \in \mathcal{C}(\mathcal{G}_2) \subseteq C_2$), and the cost function achieves its minimum at this point. This result is concerned with (A_1^*, A_2^*) being the *unique* optimal solution.

For one direction suppose that $T_{C_1}(A_1^*) \cap -T_{C_2}(A_2^*) = \{0\}$. Then there exists no $Z_1 \in T_{C_1}(A_1^*), Z_2 \in T_{C_2}(A_2^*)$ such that $Z_1 + Z_2 = 0$ with $Z_1 \neq 0, Z_2 \neq 0$. Consequently every feasible direction from (A_1^*, A_2^*) into $C_1 \times C_2$ would increase the value of the objective. Thus (A_1^*, A_2^*) is the unique optimum of (11).

For the other direction suppose that (A_1^*, A_2^*) is the unique optimum of (11), and assume for the sake of a contradiction that $T_{C_1}(A_1^*) \cap -T_{C_2}(A_2^*)$ contains a nonzero element, which we'll denote by Z. There exists a scalar $\alpha > 0$ such that $A_1^* + \alpha Z \in C_1$ and $A_2^* - \alpha Z \in C_2$. Consequently $(A_1^* + \alpha Z, A_2^* - \alpha Z)$ is also a feasible solution that achieves the lowest possible cost of zero. This contradicts the assumption that (A_1^*, A_2^*) is the unique optimum. $\square$

Thus we have that *transverse intersection* of the tangent cones $T_{C_1}(A_1^*)$ and $-T_{C_2}(A_2^*)$ is equivalent to *exact recovery* of (A_1^*, A_2^*) given the sum $A = A_1^* + A_2^*$. As $\mathcal{C}(\mathcal{G}_1) \subseteq C_1$ and $\mathcal{C}(\mathcal{G}_2) \subseteq C_2$, we have that $T_{\mathcal{C}(\mathcal{G}_1)}(A_1^*) \subseteq T_{C_1}(A_1^*)$ and $T_{\mathcal{C}(\mathcal{G}_2)} \subseteq T_{C_2}(A_2^*)$. These relations follow from the fact that the set of feasible directions from A_1^* and A_2^* into the respective convex sets is enlarged. Therefore the tangent cone transversality condition of Proposition 5.3 is generally more difficult to satisfy if we use relaxations C_1, C_2 to the convex hulls $\mathcal{C}(\mathcal{G}_1), \mathcal{C}(\mathcal{G}_2)$. Consequently we have a *tradeoff* between the complexity of solving the convex program, and the possibility of exactly recovering (A_1^*, A_2^*). However the following example suggests that it is possible to obtain tractable relaxations that still allow for perfect recovery.

Example. We consider the 16-cycle, the Shrikhande graph, and the Clebsch graph (see Figure 3), and investigate the deconvolution problem for all three pairings of these graphs. For illustration purposes suppose A_1^* is an adjacency matrix of the unweighted 16-node cycle denoted $\mathcal{G}_1$, and that A_2^* is an adjacency matrix of the 16-node Clebsch graph denoted $\mathcal{G}_2$ (see Figure 1). These adjacency matrices are random instances chosen from the set of all valid adjacency matrices that represent the graphs $\mathcal{G}_1, \mathcal{G}_2$. Given the sum $A = A_1^* + A_2^*$, we construct convex constraint sets C_1, C_2 as follows:

$$
\begin{aligned}
C_1 &= \mathcal{A} \cap \mathcal{E}(A_1^*) \\
C_2 &= \mathcal{A} \cap \mathcal{E}(A_2^*).
\end{aligned}
$$

Underlying graphs	# successes in 100 random trials
The 16-cycle and the Clebsch graph	100
The 16-cycle and the Shrikhande graph	96
The Clebsch graph and the Shrikhande graph	94

Table 1: A summary of the results of graph deconvolution via convex optimization: We generated 100 random instances of each deconvolution problem by randomizing over the labelings of the components. The convex program uses only spectral invariants to characterize the convex hulls of the component graphs, as described in Section 5.2.

Here $\mathcal{E}(A)$ represents the spectral constraints of Section 3.4. Therefore the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are characterized purely by their spectral properties. By running the convex program described above for 100 random choices of labelings of the vertices of the graphs $\mathcal{G}_1, \mathcal{G}_2$, we obtained *exact* recovery of the adjacency matrices (A_1^*, A_2^*) in all cases (see Table 1). *Thus we have exact decomposition based only on convex spectral constraints, in which the only invariant information used to characterize the component graphs $\mathcal{G}_1, \mathcal{G}_2$ are the spectra of $\mathcal{G}_1, \mathcal{G}_2$.* Similarly successful decomposition results using only spectral invariants are also seen in the cycle/Shrikhande graph deconvolution problem, and the Clebsch graph/Shrikhande graph deconvolution problem; Table 1 gives complete results.

The inspiration for using the Clebsch graph and the Shrikhande graph as examples for deconvolution is based on Proposition 5.3. Specifically, a graph for which the tangent cone with respect to the corresponding spectral constraint set $\mathcal{E}(A)$ (defined in Section 3.4) is small is well-suited to being deconvolved from other graphs using spectral invariants. This is because the tangent cone being smaller implies that the transversality condition of Proposition 5.3 is easier to satisfy. In order to obtain small tangent cones with respect to spectral constraint sets, we seek graphs that have many *repeated eigenvalues.* *Strongly regular graphs*, such as the Clebsch graph and the Shrikhande graph, are prominent examples of graphs with repeated eigenvalues as they have only three distinct eigenvalues. A strongly regular graph is an unweighted regular graph (i.e., each node has the same degree) in which every pair of adjacent vertices have the same number of common neighbors, and every pair of non-adjacent vertices have the same number of common neighbors [21]. We explore in more detail the properties of these and other graph classes in a separate report [12], where we characterize families of graphs for which the transverse intersection condition of Proposition 5.3 provably holds for constraint sets C_1, C_2 constructed using tractable graph invariants.

5.3 Application: Generating Graphs with Desired Properties

We first consider the problem of constructing a graph with certain desired structural properties.

Problem 2. *Suppose we are given structural constraints on a graph in terms of a collection of (possibly nonconvex) graph invariants $\{h_j(A) = \alpha_j\}$. Can we recover a graph that is consistent with these constraints? For example we may be given constraints on the spectrum, the degree distribution, the girth, and the MAXCUT value. Can we construct some graph $\mathcal{G}$ that is consistent with this knowledge?*

This problem may be infeasible in that there may no graph consistent with the given information. We do not address this feasibility question here, and instead focus only on the computational problem of generating graphs that satisfy the given constraints assuming such graphs do exist. Next we propose a convex programming approach using invariant convex sets to construct a graph $\mathcal{G}$, specified by an adjacency matrix A, which satisfies the required constraints. Both the problem as well the solution can be suitably modified to include inequality constraints.

Solution 2. *We combine information from all the invariants to construct an invariant convex set C. Given a constraint of the form $h_j(A) = \alpha_j$, we consider the following convex set:*

$$C_j = \text{conv}\{A : A \in \mathbf{S}^n, \ h_j(A) = \alpha_j\}.$$

This set is convex by construction, and is an invariant convex set if h_j is a graph invariant. If h_j is a convex graph invariant this set is equal to the sublevel set $\{A : A \in \mathbf{S}^n, \ h_j(A) \leq \alpha_j\}$. Given a collection of constraints $\{h_j(A) = \alpha_j\}$ we then form an invariant convex constraint set as follows:

$$C = \cap_j \ C_j.$$

Therefore any invariant information that is amenable to approximation as a convex constraint set can be incorporated in such a framework. For example constraints on the degree distribution or the spectrum can be naturally relaxed to tractable convex constraints, as described in Section 3.4. If the set C as defined above is intractable to compute, one may further relax C to obtain efficient approximations. In many cases of interest a subset of the boundary of C corresponds to points at which all *the constraints are active $\{A : h_j(A) = \alpha_j\}$. In order to recover one of these extreme points, we maximize a* random *linear functional defined by $M \in \mathbf{S}^n$ (with the entries in the upper-triangular part chosen to be independent and identically distributed to zero-mean, variance-one standard Gaussians) over the set C:*

$$\hat{A} = \arg\max_{A \in \mathbf{S}^n} \quad \text{Tr}(MA) \tag{12}$$
$$s.t. \quad A \in C.$$

This convex program is successful if $\hat{A}$ is indeed an extreme point at which all the constraints $\{h_j(A) = \alpha_j\}$ are satisfied.

Clearly this approach is well-suited for constructing constrained graphs only if the convex set C described in the solution scheme contains many extreme points at which all the constraints are satisfied. The next result gives conditions under which the convex program recovers an $\hat{A}$ that satisfies all the given constraints:

Proposition 5.4. *Consider the problem and solution setup as defined above. Define the set $\mathcal{N}$ as follows:*

$$\mathcal{N} = \bigcup_{\{A \ : \ A \in C, \ h_j(A) = \alpha_j \ \forall j\}} \mathcal{N}_C(A).$$

If $M \in \mathcal{N}$ then the optimum $\hat{A}$ of the convex program (12) *satisfies all the specified constraints exactly. In particular if M is chosen uniformly at random as described above, then the probability of success is equal to the fraction of $\mathbf{S}^n$ covered by the union of the normal cones $\mathcal{N}$.*

Proof. The proof follows from standard results in convex analysis. In particular we appeal to the fact that a linear functional defined by M achieves its maximum at $\hat{A} \in C$ if and only if $M \in \mathcal{N}_C(\hat{A})$. □

As a corollary of this result we observe that if the invariant information provided exactly characterizes the convex hull of a graph $\mathcal{G}$, then the set C above is the convex hull $\mathcal{C}(\mathcal{G})$ of the graph $\mathcal{G}$. In such cases the convex program given by (12) produces an adjacency matrix representing $\mathcal{G}$ with probability one. Next we provide the results of a simple experiment that demonstrates the effectiveness of our approach in generating sparse graphs with large spectral gap.

Figure 4: An adjacency matrix of a sparse, well-connected graph example obtained using the approach described in Section 5.3: The weights of this graph lie in the range $[0,1]$, the black points represent edges with nonzero weight, and the white points denote absence of edges. The (weighted) degree of each node is 8, the average number of nonzero (weighted) edges per node is 8.4, the second-smallest eigenvalue of the Laplacian is 4, and the weighted diameter is 3.

Example. In this example we aim to construct graphs on $n = 40$ nodes with adjacency matrices in $\mathcal{A}$ that have degree $d = 8$, node weights equal to zero, and the second-smallest eigenvalue of the Laplacian being larger than $\epsilon = 4$. The goal is to produce relatively *sparse* graphs that satisfy these constraints. The specified constraints can be used to construct a convex set as follows:

$$C = \{A : A \in \mathcal{A}, \ \tfrac{1}{8}A\mathbf{1} = \mathbf{1}, \ \lambda_{n-1}(L_A) \geq 4, \ A_{ii} = 0 \ \forall i\}.$$

By maximizing 100 random linear functionals over this set we obtained graphs in all 100 cases with total degree equal to 8, and in 98 of the 100 cases with the minimum eigenvalue of the Laplacian equal to 4 (it is greater than 4 in the remaining two cases). Interestingly the average number of edges with nonzero weight incident on each node is 8.8 over these 100 trials, thus providing very sparse graphs that are well-connected. Figure 4 gives an example of a graph generated randomly using this procedure; the average number of nonzero (weighted) edges per node of this graph is 8.4, and its (weighted) diameter is 3. Therefore this approach empirically yields sparse graphs that are well-connected (i.e., with a large spectral gap).

We would like to point out here a different approach to constructing well-connected graphs, which tries to add edges from a subset of candidate edges to maximize the second eigenvalue of the graph Laplacian [20]. An interesting question is to understand the structure of the extreme points of the set C in this example as the graph size and the degree (n, d) grow large, with ϵ held constant. For example it may be useful to compute the fraction of the normal cones at those extreme points corresponding to expander graphs. More generally it is of interest to give conditions on constraint sets under which the procedure described above is successful in providing graphs that satisfy all the constraints with high probability.

5.4   Application: Graph Hypothesis Testing

Finally we give a solution to the hypothesis testing problem in which we have two families of graphs, and the goal is to decide which of these families offers a "better explanation" for a given candidate "sample" graph.

Problem 3. *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ denote two families of graphs characterized in terms of invariants $\{h_j^1\}$ and $\{h_j^2\}$ respectively; for example, a family could be specified as some set of graphs that have similar spectral distributions, similar degree sequences, and similar girths. Given a graph $\mathcal{G}$, which of the two families $\mathcal{F}_1, \mathcal{F}_2$ of graphs is more similar to $\mathcal{G}$?*

We emphasize that the sets of invariants that characterize $\mathcal{F}_1, \mathcal{F}_2$ may in general be very different. Note that this question is not completely well-posed, as there may be different answers depending on one's notion of similarity. In order to address this point, we need to develop a statistical theory for graphs. In such a setting one could phrase this question formally as a statistical hypothesis testing problem with appropriate error metrics. Our focus in the present paper is on proposing a convex optimization solution to the hypothesis testing based on convex graph invariants, and using a reasonable notion of similarity.

Solution 3. *Let $A \in \mathbf{S}^n$ be an adjacency matrix that represents the graph $\mathcal{G}$. We construct invariant convex sets C_1 and C_2 based on the sets of invariants $\{h_j^1\}, \{h_j^2\}$ in an analogous manner to the construction described in the solution to the graph construction problem of Section 5.3. As before one could employ further tractable relaxations of these sets if they are intractable to compute. Assuming that these convex constraint sets that summarize the families $\mathcal{F}_1$ and $\mathcal{F}_2$ are compact, we declare that $\mathcal{F}_1$ is closer to $\mathcal{G}$ than $\mathcal{F}_2$ if the following holds:*

$$\max_{M \in C_1} \; \mathrm{Tr}(AM) \quad \geq \quad \max_{M \in C_2} \; \mathrm{Tr}(AM). \tag{13}$$

Naturally we declare the opposite result if the inequality is switched. Computing the two sides in this test can be done via convex optimization, and this computation is tractable if C_1, C_2 are tractable to characterize.

Our choice of the function to be maximized over C_1, C_2 is motivated by a similar procedure in statistics and signal processing, which goes by the name of "matched filtering." Of course other (convex invariant) cost functions can also be optimized depending on one's notion of similarity. We point out two advantages of this approach to hypothesis testing. First the two families of graphs can be specified in terms of different sets of invariants, as seen in these examples. Second the optimal solutions of the convex programs in (13) in fact provide *approximations* to the graph $\mathcal{G}$ by elements in the families $\mathcal{F}_1, \mathcal{F}_2$. We give illustrations of these points in our examples, which we describe next.

Example. Let A_{cycle} denote the adjacency matrix of a 16-node unweighted cycle. As our first family we consider the set of cycles on 16 nodes. We approximate this family by the set of graphs that are triangle-free (in the sense described in Section 3.4), have degree equal to 2, and have the same spectrum as a 16-node unweighted cycle. Therefore the set C_1 is defined as follows:

$$C_1 = \{A : A \in \mathcal{A}, \; A_{ii} = 0 \; \forall i, \; \tfrac{1}{2}A\mathbf{1} = \mathbf{1}, \; \Theta_{K_3}(A) \leq 4\} \cap \mathcal{E}(A_{\mathrm{cycle}}).$$

As our second family, we consider sparse well-connected graphs on 16 nodes with maximum weighted degree less than or equal to 2.5, and with the second-smallest eigenvalue of the Laplacian bounded below by 1.1:

$$C_2 = \{A : A \in \mathcal{A}, \; A_{ii} = 0 \; \forall i, \; (A\mathbf{1})_i \leq 2.5 \; \forall i, \; \lambda_{n-1}(L_A) \geq 1.1\}.$$

Applying the solution described above to a test graph given by a 16-node unweighted path graph (i.e., an unweighted cycle with an edge removed, see Figure 2), we find that the path graph is "closer" to the family $\mathcal{F}_1$ of cycles approximated by the set C_1 than it is to the family $\mathcal{F}_2$. This

agrees with the intuition that a path graph is not well-connected, and is only one edge away from being a cycle. We also point out that the optimal solution to the convex program on the left-hand-side of the test (13) is in fact an unweighted 16-node cycle with the missing edge in the path graph added as an extra edge. Next we consider a different test graph – a 16-node cycle with two additional edges across diametrically opposite nodes, i.e., assuming we label the nodes of the 16-node cycle we add edges between nodes 1 and 9, and between nodes 5 and 13 (again see Figure 2). While this graph is only two edges away from being a cycle, the edges connecting far away nodes dramatically increase the connectivity of the graph. In this case we find using the convex programming hypothesis test (13) that the family $\mathcal{F}_2$ is in fact closer than $\mathcal{F}_1$ to the sample graph. Interestingly, the optimal solution to the convex program on the left-hand-side of the test (13) is again an unweighted 16-node cycle, this time with the two additional edges removed.

In order to thoroughly address the graph hypothesis testing problem, we need to develop a framework of statistical models over spaces of graphs. With a proper statistical framework in place we can evaluate the *probability of error* achieved by a hypothesis-testing algorithm with respect to a suitable error-metric, analogous to similar methods developed in other classical decision-theoretic problems in statistics. We defer these questions to a separate paper.

6   Discussion

In this paper we introduced and studied convex graph invariants, which are graph invariants that are convex functions of the adjacency matrix. Convex invariants form a rich subset of the set of all graph invariants, and they are useful in developing a unified computational framework based on convex optimization to solve a number of graph problems. In particular we described three canonical problems involving the structural properties of graphs, namely, graph construction given constraints, graph deconvolution of a composite graph into individual components, and graph hypothesis testing in which the objective is decide which of two given families of graphs offers a better explanation for a new sample graph. We presented convex optimization solutions to all of these problems, with convex graph invariants playing a prominent role. These solutions provided attractive empirical performance, and the resulting convex programs are tractable and can be solved using general-purpose off-the-shelf software for moderate size instances.

We are presently investigating several research questions arising from this paper. It is of interest to provide theoretical guarantees on the performance of our convex programs in Section 5 in solving the problems of Section 2. For example which families of graphs can be deconvolved or efficiently sampled from using convex optimization? It is also preferable to develop special-purpose software to efficiently compute some subset of convex graph invariants, in order to enable the solution of very large problem instances. Finally in order to properly analyze the success of algorithms for graph deconvolution, sampling, and hypothesis testing, it is important to develop a formal statistical framework for graphs.

A   Properties of Convex Symmetric Functions

A *convex symmetric function* is a convex function that is invariant with respect to a permutation of the argument:

Definition A.1. *A function* $g : \mathbb{R}^n \to \mathbb{R}$ *is a* convex symmetric function *if it is convex, and if for any* $\mathbf{x} \in \mathbb{R}^n$ *it holds that* $g(\Pi\mathbf{x}) = g(\mathbf{x})$ *for all permutation matrices* $\Pi \in \mathrm{Sym}(n)$.

The properties of such functions are well-known in the literature on convex analysis and optimization, and they arise in many applications. We briefly describe some of these properties and applications here.

An important class of convex symmetric functions is the set of linear functionals given by *monotone linear functionals*:

$$g(\mathbf{x}) = \mathbf{v}^T \overline{\mathbf{x}},$$

where $\mathbf{v}_1 \geq \cdots \geq \mathbf{v}_n$. Recall that $\overline{\mathbf{x}}$ is the vector obtained by sorting the entries of $\mathbf{x}$ in descending order. Monotone linear functionals can be used to express any convex symmetric function. Specifically, let $\mathcal{M} \subset \mathbb{R}^n$ represent the cone of monotone decreasing vectors in $\mathbb{R}^n$. Then for any convex symmetric function $g : \mathbb{R}^n \to \mathbb{R}$, we have that

$$g(\mathbf{x}) = \sup_{\mathbf{v} \in \mathcal{M}} \ \mathbf{v}^T \overline{\mathbf{x}} - \alpha_{\mathbf{v}}.$$

This statement is a simple consequence of the separation theorem from convex analysis [35]. Monotone linear functionals in turn can be expressed as the nonnegative *sum* of even more elementary functions called *distribution functions*, which are defined as follows:

$$g_k(\mathbf{x}) = \sum_{i=1}^{k} \ (\overline{\mathbf{x}})_i.$$

These functions are closely related to the notion of conditional value-at-risk [36], which in turn is computed using quantiles of probability distributions.

Convex symmetric functions are intimately connected with the concept of *majorization* [30]. There are many equivalent characterizations of majorization [13, 28], and we briefly mention some of these next. A vector $\mathbf{x} \in \mathbb{R}^n$ is said to majorize another vector $\mathbf{y} \in \mathbb{R}^n$ if

$$g_k(\mathbf{x}) \geq g_k(\mathbf{y}), \ \forall k = 1, \ldots, n-1 \quad \text{and} \quad g_n(\mathbf{x}) = g_n(\mathbf{y}).$$

The *permutahedron* of a vector $\mathbf{x} \in \mathbb{R}^n$ is the convex hull of all permutations of $\mathbf{x}$, and is given by the set of vectors in $\mathbb{R}^n$ that are majorized by $\mathbf{x}$. Thus, convex constraints given by distribution functions provide a simple characterization of the permutahedron generated by $\mathbf{x}$. Majorization is also closely related to the notion of *Lorenz dominance*; a (typically nonnegative) vector $\mathbf{x} \in \mathbb{R}^n$ is said to Lorenz-dominate $\mathbf{y} \in \mathbb{R}^p$ if $-\mathbf{x}$ is majorized by $-\mathbf{y}$. Lorenz dominance is used to measure the level of inequality in distributions, i.e., if a distribution $\mathbf{x}$ Lorenz-dominates a distribution $\mathbf{y}$ then $\mathbf{x}$ is "more equal" than $\mathbf{y}$ (see also the Gini coefficient, which is used to measure inequalities in countries).

A convex symmetric function is an example of a *Schur-convex function*, which is a function f such that $f(\mathbf{x}) \geq f(\mathbf{y})$ whenever $\mathbf{x}$ majorizes $\mathbf{y}$. Hence a Schur-convex function preserves order with respect to majorization. Consequently, such functions arise in many applications in which majorization plays a prominent role [30]. We note that the functions that are both convex and Schur-convex are exactly the convex symmetric functions.

A fairly similar set of results hold for convex functions of symmetric matrices that are invariant under conjugation of the argument by orthogonal matrices, i.e., convex functions $f : \mathbf{S}^n \to \mathbb{R}$ such that $f(VAV^T) = f(A)$ for all $A \in \mathbf{S}^n$ and for all $\Pi \in \mathrm{Sym}(n)$.

References

[1] AMES, B. AND VAVASIS, S. (2009). Nuclear norm minimization for the planted clique and biclique problems. *Preprint.* arXiv:0901.3348.

[2] AMES, B. AND VAVASIS, S. (2010). Convex optimization for the planted k-disjoint-clique problem. *Preprint.* arXiv:1008.2814.

[3] ARTZNER, P., DELBAEN, F., EBER, J., AND HEATH, D. (1999). Coherent measures of risk. *Math. Fin.* **9** 203–228.

[4] BENTAL, A., EL GHAOUI, L., AND NEMIROVSKI, A. (2009). *Robust Optimization.* Princeton University Press.

[5] BENTAL, A. AND NEMIROVSKI, A. (2001). *Lectures on Modern Convex Optimization.* SIAM.

[6] BERTSIMAS, D. AND BROWN, D. (2009). Constructing Uncertainty Sets for Robust Linear Optimization. *Oper. Res.* **57** 1483–1495.

[7] BIGGS, N. (1994). *Algebraic Graph Theory.* Cambridge University Press.

[8] BONCHEV, D. (1991). *Chemical Graph Theory: Introduction and Fundamentals.* Taylor and Francis.

[9] BOYD, S. (2006). Convex Optimization of Graph Laplacian Eigenvalues. *Proc. Int'l. Cong. of Math.* **3** 1311–1319.

[10] BOYD, S., DIACONIS, P., AND XIAO, L. (2004). Fastest Mixing Markov Chain on a Graph. *SIAM Rev.* **46** 667–689.

[11] CELA, E. (1998). *The Quadratic Assignment Problem: Theory and Algorithms.* Springer.

[12] CHANDRASEKARAN, V., PARRILO, P., AND WILLSKY, A. Graph Deconvolution via Semidefinite Programming. *In preparation.*

[13] DAVIS, D. (1957). All convex invariants of Hermitian matrices. *Arch. der Math.* **8** 276–278.

[14] DE KLERK, E. AND SOTIROV, R. (2010). Exploiting group symmetry in semidefinite programming relaxations of the Quadratic Assignment Problem. *Math. Prog.*, **122** 225–246.

[15] DIESTEL, R. (2005). *Graph Theory.* Springer.

[16] DOBRIN, R., BEG, Q., BARABASI, A., AND OLTVAI, Z. (2004). Aggregation of topological motifs in the Escherichia coli transcriptional regulatory network. *BMC Bioinf.* **5**.

[17] DODZIUK, J. (1984). Difference equations, isoperimetric inequality and transience of certain random walks. *Trans. Amer. Math. Soc.* **284** 787-794.

[18] EASLEY, D. AND KLEINBERG, J. (2010). *Networks, Crowds, and Markets: Reasoning about a Highly Connected World.* Cambridge University Press.

[19] FINKE, G., BURKARD, R., AND RENDL, F. (1987). Quadratic Assignment Problems. *Ann. of Disc. Math.* **31** 61–82.

[20] GHOSH, A. AND BOYD, S. (2006). Growing Well-Connected Graphs. *Proc. IEEE Conf. on Dec. and Cont.* 6605-6611.

[21] GODSIL, C. AND ROYLE, G. (2004). *Algebraic Graph Theory.* Springer-Verlag.

[22] Goemans, M. and Williamson, D. (1995). Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Jour. of the ACM.* **42** 1115–1145.

[23] Goldreich, O., Goldwasser, S., and Ron, D. (1996). Property testing and its connection to learning and approximation. *Proc. of Ann. Symp. on Foun. of Comp. Sci.*.

[24] Hoory, S., Linial, N., and Wigderson, A. (2006). Expander Graphs and their Applications. *Bull. Amer. Math. Soc.* **43** 439–561.

[25] Jackson, M. (2008). *Social and Economic Networks.* Princeton University Press.

[26] Kleinberg, J. and Tardos, E. (2008). Balanced Outcomes in Social Exchange Networks. *Proc. Symp. on Theo. of Comp.*

[27] Lauritzen, S. L. (1996). *Graphical models.* Oxford University Press.

[28] Lewis, A. (1995). The Convex Analysis of Unitarily Invariant Matrix Functions. *Jour. of Convex Analy.* **2** 173–183.

[29] Löfberg, J. (2004). YALMIP: A Toolbox for Modeling and Optimization in MATLAB. *Proceedings of the CACSD Conference, Taiwan.* Available from http://control.ee.ethz.ch/ joloef/yalmip.php.

[30] Marshall, A. and Olkin, I. (1979). *Inequalities: The Theory of Majorizations and Its Applications.* Academic Press.

[31] Mason, O. and Verwoerd, M. (2007). Graph Theory and Networks in Biology. *IET Syst. Biol.* **1** 89-119.

[32] Motzkin, T. and Straus, E. (1965). Maxima for graphs and a new proof of a theorem of Turan. *Canad. J. Math.* **17** 533–540.

[33] Rendl, F. and Sotirov, R. (2007). Bounds for the Quadratic Assignment Problem using the bundle method. *Math. Prog.* **109** 505–524.

[34] Robertson, N. and Seymour, P. (1984). Graph minors III: Planar tree-width. *Jour. of Comb. Theo., Series B.* **36** 49-64.

[35] Rockafellar, R. (1970). *Convex Analysis.* Princeton University Press.

[36] Rockafellar, R. and Uryasev, S. (2000). Optimization of Conditional Value-at-Risk. *Jour. of Risk.* **2** 21–41.

[37] Rudin, W. (1966). *Real and Complex Analysis.* McGraw-Hill.

[38] Toh, K., Todd, M., and Tutuncu, R.. *SDPT3 - a MATLAB software package for semidefinite-quadratic-linear programming.* Available from http://www.math.nus.edu.sg/ mattohkc/sdpt3.html.

[39] Watson, G. A. (1992). Characterization of the subdifferential of some matrix norms. *Linear Algebra and Applications.* **170** 1039–1053.

[40] Zhao, Q., Karisch, S., Rendl, F., and Wolkowicz, H. (1998). Semidefinite Programming Relaxations for the Quadratic Assignment Problem. *Jour. of Comb. Opt..* **2** 71–109.

[41] Ziegler, G. (1995). *Lectures on Polytopes.* Springer.